SPN Documentation

Release

yguan

Aug 05, 2017

Contents

1 Tutorials 3

2 User's Guide 5

3 API Reference 7

4 Indices and tables 13

Python Module Index 15

SPN is library to build, train and save neural networks based on Theano.

SPN defines a neural network image on hard disk to reuse and modify.

CHAPTER 1

Tutorials

Traning Neural Networks on MNIST

This tutorial goes through MNIST digits problem to explain the usage of SPN.

Preparing the Data

Defining a MLP Network

CHAPTER 2

User's Guide

CHAPTER 3

API Reference

The following is the document extracted from code.

Network

class mlbase.network.**Network**

Theano based neural network.

build(*reload=False*)

Build the training function and predict function after collecting all the necessary information.

buildForwardSize()

Initialize parameter based on size info for each layer.

connect(*prelayer*, *nextlayer*, *reload=False*)

Connect prelayer to nextlayer.

getLastLinkName()

Get last link file name, including path prefix.

getSaveModelName(*dateTime=None*)

Return default model saving file name, including path prefix.

load(*istream*)

Load the model from input stream. reset() is called to clean up network instance.

nextLayer()

Use this method to iterate over all known layers. This is a DAG walker. Guarantee all previous layers are visited for the next visiting layer.

reset()

For sequential layerout network, use append().

To add more layers, the first layer is set with setInput(). Network can do this, because it remember which layer to append to by using member variable currentLayer.

save (*ostream*)
Save model to the stream.

saveToFile (*fileName=None*)
Save the network to a file. Use the given file name if supplied. This may take some time when the model is large.

updateLatestLink ()
Create sym link to leates saved model.

mlbase.layers

Network input

class mlbase.layers.**RawInput** (*inputsize*)
This is THE INPUT Class. Class type is checked during network building.

Parameters **input** (*tuple or list of inte*) – Input shape without batch size.

setBatchSize (*psize*)
This method is suposed to called by network.setInput()

class mlbase.layers.**NonLinear**

predictForward (*inputtensor*)
forward link used in training

inputtensor: a tuple of theano tensor

return: a tuple of theano tensor

class mlbase.layers.**Relu**

class mlbase.layers.**Elu** (*alpha=1.0*)

class mlbase.layers.**ConcatenatedReLU**

class mlbase.layers.**Sine**

class mlbase.layers.**Cosine**

class mlbase.layers.**SeqLayer** (*name*, *bases*, *namespace*, ***kwds*)

class mlbase.layers.**DAGPlan**

classmethod **input** ()
Intend to be the input for the layers.

class mlbase.layers.**DAG** (*name*, *bases*, *namespace*, ***kwds*)

<no title>

NonLinear
Relu
Elu

Continued on next page

Table 3.1 – continued from previous page

ConcatenatedReLU
Sine
Cosine

<no title>

<no title>

SeqLayer
DAGPlan
DAG

<no title>

<no title>

<no title>

<no title>

<no title>

<no title>

<no title>

Network input

RawInput	This is THE INPUT Class.

<no title>

<no title>

<no title>

Cost function

class mlbase.cost.**CostFunc**
General cost function base class.

Y: result from forward network. tY: the given true result.

class mlbase.cost.**TwoStageCost**
Cost function that needs two stage computation.

Step 1: obtain data statistics. Step 2: obtain label for each sample.

class mlbase.cost.**IndependentCost**
Cost function for each sample cost known and final cost is a statistics for all sample cost.

mlbase.cost.**aggregate**(*loss*, *weights=None*, *mode='mean'*)
This code is from lasagne/objectives.py

class mlbase.cost.**CrossEntropy**
Wrap of categorical_crossentropy from theano

class mlbase.cost.**ImageDiff**
This is the base class for cost function for images. The input format is like:

tensor4, (patch, channel, column, row)

The channel should be 1 or 3.

class mlbase.cost.**ImageSSE**
The sum of square error. Use aggregate() to get mean square error.

class mlbase.cost.**ImageDice**
Dice coefficient. Y is the set of salient pixel in one binary image tY is another set of salient pixel in the other binary image. The Dice coefficient is: 2 * |Y ^ tY| / (|Y| + |tY|)

Optimizer

class mlbase.gradient_optimizer.**GradientOptimizer**(*lr*)

class mlbase.gradient_optimizer.**RMSprop**(*lr=0.01*, *rho=0.9*, *epsilon=1e-06*)

class mlbase.gradient_optimizer.**Adam**(*lr=0.01*, *beta1=0.9*, *beta2=0.999*, *epsilon=1e-07*)

class mlbase.gradient_optimizer.**Momentum**(*lr=0.01*, *mu=0.5*)

class mlbase.gradient_optimizer.**Nesterov**(*lr=0.01*, *mu=0.5*)

class mlbase.gradient_optimizer.**Adagrad**(*lr=0.01*, *epsilon=1e-07*)

Regularization

class mlbase.regularization.**Regulator**(*weight_decay=0.0005*, *reg_func=<function l2>*)
Regulator added to cost function.

mlbase.regularization.**l1**(*x*)
L1 penalty

mlbase.regularization.**l2**(*x*)
L2 penalty

Utility

class mlbase.layers.**RawInput**(*inputsize*)
This is THE INPUT Class. Class type is checked during network building.

Parameters **input** (*tuple or list of inte*) – Input shape without batch size.

setBatchSize(*psize*)
This method is suposed to called by network.setInput()

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mlbase.cost, 10
mlbase.gradient_optimizer, 11
mlbase.layers, 8
mlbase.layers.activation, 8
mlbase.layers.compose, 8
mlbase.layers.rawinput, 11
mlbase.network, 7
mlbase.regularization, 11

Index

A

Adagrad (class in mlbase.gradient_optimizer), 11
Adam (class in mlbase.gradient_optimizer), 11
aggregate() (in module mlbase.cost), 10

B

build() (mlbase.network.Network method), 7
buildForwardSize() (mlbase.network.Network method), 7

C

ConcatenatedReLU (class in mlbase.layers), 8
connect() (mlbase.network.Network method), 7
Cosine (class in mlbase.layers), 8
CostFunc (class in mlbase.cost), 10
CrossEntropy (class in mlbase.cost), 10

D

DAG (class in mlbase.layers), 8
DAGPlan (class in mlbase.layers), 8

E

Elu (class in mlbase.layers), 8

G

getLastLinkName() (mlbase.network.Network method), 7
getSaveModelName() (mlbase.network.Network method), 7
GradientOptimizer (class in mlbase.gradient_optimizer), 11

I

ImageDice (class in mlbase.cost), 10
ImageDiff (class in mlbase.cost), 10
ImageSSE (class in mlbase.cost), 10
IndependentCost (class in mlbase.cost), 10
input() (mlbase.layers.DAGPlan class method), 8

L

l1() (in module mlbase.regularization), 11
l2() (in module mlbase.regularization), 11
load() (mlbase.network.Network method), 7

M

mlbase.cost (module), 10
mlbase.gradient_optimizer (module), 11
mlbase.layers (module), 8
mlbase.layers.activation (module), 8
mlbase.layers.compose (module), 8
mlbase.layers.rawinput (module), 8, 11
mlbase.network (module), 7
mlbase.regularization (module), 11
Momentum (class in mlbase.gradient_optimizer), 11

N

Nesterov (class in mlbase.gradient_optimizer), 11
Network (class in mlbase.network), 7
nextLayer() (mlbase.network.Network method), 7
NonLinear (class in mlbase.layers), 8

P

predictForward() (mlbase.layers.NonLinear method), 8

R

RawInput (class in mlbase.layers), 8, 11
Regulator (class in mlbase.regularization), 11
Relu (class in mlbase.layers), 8
reset() (mlbase.network.Network method), 7
RMSprop (class in mlbase.gradient_optimizer), 11

S

save() (mlbase.network.Network method), 7
saveToFile() (mlbase.network.Network method), 8
SeqLayer (class in mlbase.layers), 8
setBatchSize() (mlbase.layers.RawInput method), 8, 11
Sine (class in mlbase.layers), 8

T

TwoStageCost (class in mlbase.cost), 10

U

updateLatestLink() (mlbase.network.Network method), 8